

The bioinformatics lab SS11

Mail / DNS

Antonia Stank

20.06.2011

1 Introduction

The Domain Name System (DNS) is a one of the most important systems in the network. It is a hierarchical naming system which is built on a distributed database for computers, services or any resource connected to the Internet. The DNS can also be used in a private or local network.

The main service of a DNS is that it serves as a kind of “phone book” for the internet or local network by translating user-friendly hostnames into specific IP addresses. For example it is much easier to remember the internet address: `www.google.de` than the corresponding IP address `209.85.148.99`. So it can translate names to numbers and vice versa and helps therefore to connect on a specific IP address without knowing the number.

An e-mail-server or just mail server is a server which can handle the mail traffic. It is necessary for sending, receiving, forwarding and saving mails.

An Internet Message Access Protocol (IMAP) server is a server that provides the usage of the most prevalent IMAP protocol. There is also another common protocol which is the so called Post Office Protocol (POP). The advantage of the IMAP is that it allows an e-mail client access to a remote mail server. The e-mail clients using IMAP leave the mails on the server until they are explicitly deleted by the user. This and some other advantages make it possible to allow multiple clients to have access to the same mailbox. The data in the mailboxes are synchronized and therefore more than one user can work on mails in the same mailbox. For that the possibility of using the so called “flags” is really helpful to show other clients which status a mail has (read, replied, deleted etc.).

Icedove is an open source mail client which supports IMAP and which we will use in our lab.

Our programming challenge will be to:

- set up and configure a DNS server

- set up and configure a mail server
- set up an IMAP server
- use Icedove to send mails to course members and configure the address book to connect to the LDAP server

Before starting it is necessary to install the following packages on your system:

Package name	Package description
gnome-core	The GNOME Desktop Environment - essential components. These are the core components of the GNOME Desktop environment, an intuitive and attractive desktop.
xorg	a full featured X server
icedove	Icedove is a free/unbranded thunderbird mail client.
bind9	The Berkeley Internet Name Domain (BIND) implements an internet domain name server.
dnsutils	Various client programs related to DNS that are derived from the BIND source tree.
postfix	A mail server which is fast, easy to administrate and secure.
postfix-doc	The postfix documentation (not “really” necessary).
bsd-mailx	mailx is the traditional command-line-mode mail user agent.
dovecot-imapd	Dovecot is a mail server whose major goals are security and extreme reliability. It tries very hard to handle all error conditions and verify that all data is valid, making it nearly impossible to crash. It should also be pretty fast, extensible, and portable. This package contains the dovecot IMAP server.
ca-certificates	Containing certificates which makes it possible to check the authenticity of an SSL-Connection
procmail	Can be used to create mail-servers, mailing lists, sort your incoming mail into separate folders, preprocess your mail, start any programs upon mail arrival or selectively forward certain incoming mail automatically to someone.

2 DNS

First of all you have to edit the file `/etc/bind/named.conf.local` and add the two zones which contain the relevant information about the mapping files.

```
zone "course" {
    type master;
    file "/etc/bind/db.course";
};
zone "16.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168.16";
};
```

The two files which are defined now have to be edited. The `/etc/bind/db.course` should contain the following information:

```
;
; BIND reverse data file for broadcast zone
;
$TTL      86400
@          IN      SOA      lkajan.course. root.lkajan.course. (
                                10051701      ; Serial
                                604800         ; Refresh
                                86400          ; Retry
                                2419200        ; Expire
                                86400 )        ; Negative Cache TTL
;
@          IN      NS       lkajan.course.

lkajan     A        192.168.16.2
<other course members>
```

It is obvious that you have to change the user `lkajan` into your user name and also the IP address. After that you can add all the course members and map the name to the IP address. Another important point is that at any time when you change this file (e.g. to add another course member) you should increase the number in front of “Serial”. If you don’t do that it can happen, that the additional information gets lost. The `/etc/bind/db.192.168.16` has also to be edited and has to look as follows:

```
;
; BIND reverse data file for broadcast zone
;
$TTL      86400
@          IN      SOA      lkajan.course. root.lkajan.course. (
                                10051701      ; Serial
                                604800         ; Refresh
                                86400          ; Retry
                                2419200        ; Expire
                                86400 )        ; Negative Cache TTL
;
@          IN      NS       lkajan.course.

2          PTR      lkajan.course.
<other course members>
```

Change again the same things as in the file before and add the members by using the last number of their IP address and their host name. After you have done this you can make a test-load of your named configuration by using the following command:

```
named-checkconf -z
```

You will see all defined zones which should include the two new ones. Now you have to restart the name server by restarting “bind9”. Just use the following command `/etc/init.d/bind9 restart`. Now you have to update the `/etc/resolv.conf` file where you define your own name server:

```
search course
nameserver 127.0.0.1
...
```

Perfect, the DNS is now set up and configured. You can test it by using one of the following commands:

```
host <name>.course;
dig <name>.course
ping <name>.course
```

3 Mail server

Now we want to set up and configure the mail server. First of all we have to edit the following file `/etc/postfix/main.cf` and add our IP address to the `mynetworks` list. Restart postfix and check if the restart was successful by looking into the logs (`/var/log/syslog`). Check now if a root alias is given in the `/etc/aliases` file. If not please add:

```
postmaster:    root
root:          <your username>
```

After changing the file you have to rebuild the alias database by using the command `'newaliases'`. The last step is to edit the `~/.procmailrc` file. It's not existing yet and be careful if you are logged in as root that you edit it in your user directory. Just add the following lines with which you define the directory for incoming mails:

```
# Maildir:
DEFAULT="$HOME/Maildir/"
```

That's it! Now you can try to send a mail to yourself from root for example like this:

```
mail <your username>@<your username>.course <Enter>
Subject: Hello <Enter>
Hello,
That's just a test!
Bye!
.<Enter> or Ctrl+D
```

CC:<Enter>

You can check again your log if the mail was successfully delivered. You can read the mail when you are logged in as your user and just type 'mail'. You get a list of new mails where you should find the one you sent from root.

The procmail agent uses recipes to determine where to deliver the various mail messages. Each recipe that procmail uses consists of a mode, a condition and an action part. The recipes are configured in the ~/.procmailrc file and are read from top to bottom, meaning the order of recipes matters.

Each recipe begins with a ':0' ("colon and zero") and after that you can define which part of the mail has to be used: 'h' for header and 'b' for body or 'hb' for both. If you define nothing the header will be used as default.

Now you can define some conditions which are often given as a regular expression and are marked with an '*' at the beginning of the line. After that you can define an action, for example to save the mails with the specific conditions in a specific directory.

In our programming challenge there was an advanced one which will be used now to show how a recipe works. The task was to make an automatical reply to the sender if the mail subject contains some specific words. One of these recipes could look like this:

```
#Recipe exam & work
:0
* !^FROM_DAEMON
* !^X-Loop: stank@stank.course
* ^Subject:.*(exam)+.*(work)+
| (formail -rt -A"X-Loop: stank@stank.course"; \
    echo "I am ill!"; \
    ) | $SENDMAIL -t
```

With the given information above this recipe can be easily understood. The first line just says that a new recipe is starting and we just want as default to consider the header of the mail. The next three lines are the conditions which say that the mail should not contain lines that start with "FROM_DAEMON" and "X-Loop: stank@stank.course". Additionally it checks the line with the subject if it contains the words "exam" and "work". If all these three conditions are fulfilled the action part of the recipe starts, which means that the mail is piped to the formail program. The formail makes an automatic reply mail and adds a line containing "X-Loop: stank@stank.course" and adds the text "I am ill!". All that is again piped to \$SENDMAIL and is sent away. But what is the relevance of this "X-Loop" line? Suggesting another course member has the same recipe and you both do not have this condition with the X-Loop line, the mail will be send back and forth and will not end, so you get a loop. Therefore you give your auto reply a kind of

“fingerprint” by adding the X-Loop line and check for each incoming mail if it has already your “fingerprint”.

4 IMAP server

The specific configuration for IMAP can be made in the file `/etc/dovecot/dovecot.conf`. But as already mentioned there is nothing to do for the user now. In this file you can look up different configurations for example which protocols are used (in our case it's IMAP and IMAPS) and find many information, for example about the login process. But if you want to get more details about anything you should look into the following directory: `/usr/share/doc/dovecot-common/wiki/`.

Dovecot can use different ways to authenticate a user. It can use for example the `/etc/passwd` file or any file which is like that, the `/etc/shadow` file, the LDAP, a SQL database, or VPopMail (external software to handle virtual domains). Dovecot can use multiple password databases, so if the password doesn't match in the first database it checks in the next one which can be useful if there are virtual users and also local system users. But dovecot can also use Pluggable Authentication Modules (PAM) which is a mechanism to integrate multiple low-level authentication schemes into a high-level application programming interface (API). It allows programs that rely on authentication to be written independent of the underlying authentication scheme. So it is a central management of authentication data but it cannot be used as a user database. Applications which need an authentication of the user can send an authentication request to PAM which then returns a response (“success” or “failure”) to the application. The PAM configuration is usually in the `/etc/pam.d/` directory and dovecot uses by default `dovecot` as the PAM service name. That means you can find the PAM configuration for dovecot in `/etc/pam.d/dovecot`.

In our case it just means that dovecot sends a request to PAM if a user wants to connect to the IMAP server and allows or denies the connection. With this service the mail server is relieved because it doesn't have to deal with the authentication or any authentication files.

5 Icedove

Now we want to configure our icedove to use a user-friendly interface to manage the mails. First of all you have to use the VNC connection and the `'startx'` command to be possible to start the graphical environment of icedove. First of all you have to configure a new mail server where you have to add the following information:

Email address: `<username>@<hostname>.course`
Type: IMAP

Incoming server: <hostname>.course or 127.0.0.1
Outgoing server: <hostname>.course or 127.0.0.1

The next step is to configure LDAP by creating a LDAP directory server. Therefore go into Preferences → Composition → Addressing → Directory server → Edit directories → Add:

Hostname: localhost
Base DN: dc=course
Port n: 389
Bind DN: uid=<username>, ou=people,dc=course

Now you only have to check if your LDAP server serves connections to ldap://localhost/ in the /etc/default/slapd file and then you can send mails from icedove to other course members.

6 References

http://en.wikipedia.org/wiki/Domain_Name_System
http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol
http://de.wikipedia.org/wiki/Mail_Server
<http://lipas.uwasa.fi/~ts/info/proctips.html>
http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules
<http://www.postfix.org/>
<http://wiki2.dovecot.org/Authentication>
<http://kb.iu.edu/data/anpn.html>
<http://www.perlcode.org/tutorials/procmail/proctut/proctut1.pod>
/usr/share/doc/dovecot-common/wiki/*
man pages of the used programs